VOL. 4 NO. 7 JULY 1987

COMPUTERDIGEST

A NEW KIND OF MAGAZINE FOR ELECTRONICS PROFESSIONALS

KEYBOARDS, KEYBOARDS AND KEYBOARDS!

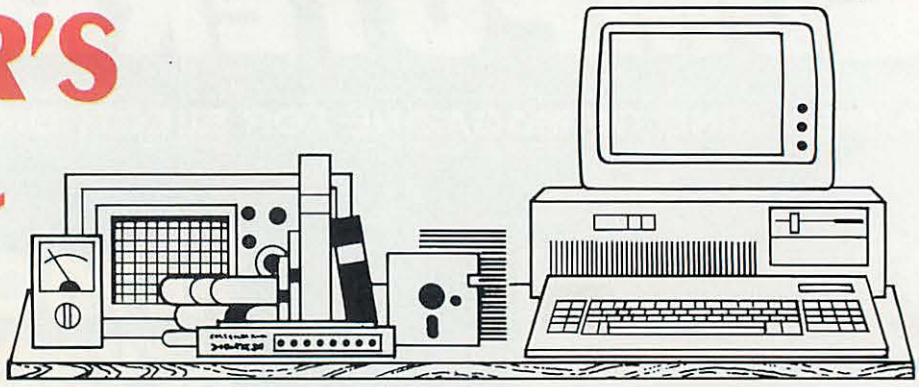
The man-machine interface

IBM's NEW PC's

First impressions of Big Blue's new entries

A GERNSBACK PUBLICATION

EDITOR'S WORK-BENCH



IBM's Personal System/2

Following months of speculation and rumor, IBM has finally released a quartet of new computers, a new version of DOS, and a slew of new peripherals, including an optical disk drive.

It's immediately apparent that the new computers are intended to open new directions for IBM, as well as to maintain continuity with past machines. And it doesn't seem that IBM has made the kind of marketing blunder it did with the PC JR and the PC Portable.

There are four new Personal System/2 computers, as shown in Fig. 1; they're dubbed the Models 30, 50, 60, and 80, and base models of each list for $1695, $3595, $5295, and $6995, respectively. As you can see, the Models 30 and 50 are desktop models, and the others are floor-standing.

The Model 30 is basically an upgraded PC (or XT); the Models 50 and 60 are basically upgraded versions of the AT; and the Model 80 is IBM's long-awaited 386 machine. The Model 30 has been rated to run about twice as fast as the PC, and the Models 50 and 60 (which differ mainly in the number of expansion slots each contains), twice as fast as the AT. The Model 80 is rated twice as fast the Models 50 and 60. Specifications are summarized in Table 1.

New features

Technical details are hard to come by at this early date, but here's what we've learned so far. (We hope to have a hands-on review next month.) The main features that distintuish the Personal System/2 computers from the old models are the new disk drives, the new video hardware, the new expansion slots, and the new unreleased operating system called Operating System/2 (OS/2).

All the new models come with 3½" disk drives. Each Model 30 disk holds 720K (twice that of a standard 5¼ inch disk); new disks for the other machines hold 1.44 megabytes each, and the Model 50, 60, and 80 disk drives can read both types of disks. The 720K disks are used in many portables currently on the market.

The new video hardware is compatible with the old CGA standard, but it also adds several new modes that are incompatible with all other standards, including IBM's own EGA as well as the Hercules standards. The new video modes offer higher resolution and more colors than the CGA, and they require new analog monitors that are incompatible with all other IBM-compatible monitors currently on the market.

The Models 30 and 50 have three expansion slots each; the Models 60 and 80 have seven slots. The slots in the Model 30 are electrically compatible with the old-style slots, but expansion slots shouldn't be necessary because the Model 30 includes most common add-on hardware: 640K of RAM, a video adapter, serial and parallel ports, mouse adapter, and a battery-backed clock/calendar. The expansion slots in the other models are totally incompatible with the old-style slots, but the new bus, which IBM calls the Micro Channel, can operate at a much higher speed than the old bus. The Models 50 and 60 have 16-bit buses; the Model 80 has a 32-bit bus.

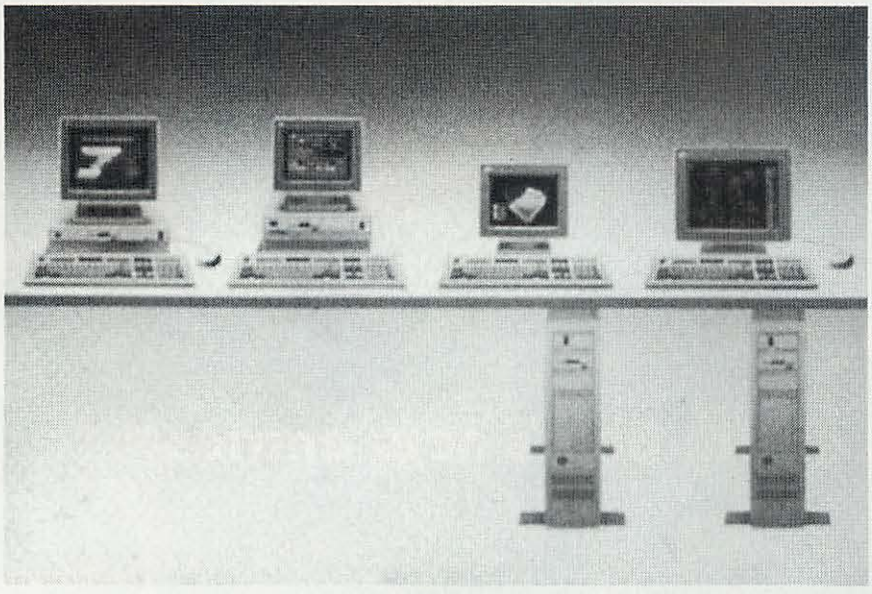


FIG. 1

DOS and OS/2

There is a new version of DOS (DOS 3.3) and a totally new operating system, which won't be released before the end of the year. DOS 3.3 has a few added commands (including a CALL statement for use in batch files), and has enhanced some old programs (BACKUP and RESTORE, for example), but all in all the new DOS does very little more than provide support for the new hardware. It runs on all past and present IBM PC's.

The other new operating system is called Operating System/2, and it contains many of the advanced features power users and network managers have been clamoring for.

OS/2 supports three "environments" and will come in three versions. The three en-

TABLE 1—IBM PERSONAL SYSTEM/2

	Model 30	Model 50	Model 60	Model 80
Microprocessor	8086	80286	80286	80386
Potential system throughput	Up to 2½ times PC XT™	Up to 2 times PersonalComputer AT®	Up to 2 times Personal Computer AT	Up to 3½ times Personal Computer AT
Standard memory Expandable to	640KB	1MB 7MB	1MB 15MB	Up to 2MB 16MB
Diskette size and capacity	3.5-inch 720KB	3.5-inch 1.44MB	3.5-inch 1.44MB	3.5-inch 1.44MB
Fixed disk Additional options	20MB	20MB	44, 70MB 44, 70, 115MB	44, 70, 115MB 44, 70, 115MB
Maximum configuration	20MB	20MB	185MB	230MB
Expansion slots	3	3	7	7
Operating system(s)	PC DOS 3.3	PC DOS 3.3 and Operating System/2™	PC DOS 3.3 and Operating System/2	PC DOS 3.3 and Operating System/2

vironments (DOS, Family, and OS/2) allow various levels of software compatiblility. The DOS environment should be totally compatible with existing programs; the OS/2 environment will allow full access to the features of the new computers (what we call the M & M's: Multi-tasking and extended Memory); and the Family environment provides a bridge between the two.

The Standard Edition Version 1.0 of OS/2 will include extensive on-line help facilities and support for the M & M's. IBM expects to release 1.0 in the first quarter of 1988. The Standard Edition Version 1.1 will include all the capabilites of 1.0, plus a Macintosh-style graphics/window user interface. IBM expects to announce a release date for 1.1 by the end of 1987. According to the rumor mill, the window interface may be Microsoft Windows, IBM's own (and neglected) Top View, or some combination thereof.

IBM calls the third version of OS/2 the Extended Edition, and it will include an advanced relational database manager, an advanced communications program (that will allow background communications), and terminal emulation. The extended edition seems to be aimed primarily at users who do a great deal of work on both PC's and mainframes.

Price and performance

It's easy to see that IBM is not going after the rock-bottom clone market, although the new models are not hopelessly expensive, either. For example, one week after IBM's announcement, Model 30's were being sold across the counter in New York City for about $1400 for the dual-floppy model and about $1800 for the floppy/hard-disk model. That price doesn't include a monitor, which runs an extra $225 or $475 (street price) for monochrome or color, respectively. However, that price does include everything IBM and the clone makers previously sold separately—video adapter card, ports, RAM, etc. Meanwhile, prices of the old models have dropped on the order of 30%, so now you can get a real IBM for the cost of a clone.

Technically speaking, the new machines indicate that 5¼" disk drives are on the way out and that 3½" drives are on the way in. The takeover will be gradual, but it is inevitable—as was the transition from 8" to 5½" disks—because the new disks are much more durable and hold much more information than the old ones.

The problem with video is much more difficult to discuss, mainly because at this early date there is little hard data. We know that the new graphics hardware is not compatible with EGA and Hercules standards, but we've been unable to find out whether it's possible to run an EGA or Hercules card in a Model 30. (The bus structure of the more powerful machines precludes EGA/Hercules use in those machines.) If it's not possible, until present-day graphics software is adapted to the new video standard, it will have to run in CGA mode. On the other hand, it appears that text-mode software will be able to take advantage of the new higher-resolution hardware, so word processors, outline processors, and the like should benefit immediately.

What to buy

If you want to buy a PC now, first you must choose between IBM and non-IBM equipment. If you choose IBM, you have to choose between old technology and the Model 30, on the one hand, and the Models 50, 60, and 80 on the other. And the choice may not be easy, depending on your needs. If you're quite sure of your present and future needs, and an old-technology machine or a Model 30 will meet those needs, buy one. Present hardware and software will power those machines for perhaps another five years without looking too dated. However, beware that software developers will gradually shift the focus of their efforts over to the new machines, and that development efforts for the old machines will gradually cease (as happened with CP/M). Choosing among a Model 30, an IBM PC (or XT), and a clone is difficult, but if I were buying today I'd lean strongly toward a Model 30 because it bridges past and future technologies.

On the other hand, if you want a machine that you can grow with, one that will be able to take advantage of the M & M's and the applications software that will put the hardware to work, buy one of the more-powerful new machines.

KEYBOARD MEDLEY

Keyboards are not all created equal. Like people, they come in a variety of sizes and shapes, and they all work differently. The differences among keyboards may seem trivial, but if you spend much time pounding on one, you'll want to ensure that it has the right feel for you. If you work on a number of different keyboards, trying to adapt to the differences among them can make you yearn for a long rest in a well-padded cell.

We want to keep you out of that cell, so here's the lowdown on various keyboards. We've got fat ones, thin ones, plain ones, fancy ones—there are eight in all, and they cover the majority of styles (PC, AT, and Enhanced) in common use. And several have features that make them attractive for special applications (typing, CAD, use by untrained users, etc.).

The original

IBM introduced the PC in 1981. The first thing many people noticed was that IBM completely abandoned the layout it had standardized on the Selectric series of typewriters. (See Fig. 1.) The major faults are that the Return key is small and hard to reach, and that the Backslash key is located between the "Z" key and the Left Shift key. Other features (which some users love, but many hate) are that the keys must be depressed far to get a response, and each depression causes the keyboard to emit a loud mechanical click (as opposed to a soft, electronically controlled one).

In addition, the Caps Lock, Num Lock, and Scroll Lock keys have no indicator lights, so you can't tell which mode you're in without typing something (and then erasing it!). Also, the layout of the numeric keypad leaves much to be desired. For example, there is no Enter key, but the "+" key occupies the space of three keys!

Key Tronic 5151

The first attempt to improve that unpopular layout was made by the Key Tronic company; it has become a standard in its own right, although it is not without its problems too.

The 5151 keyboard addressed many of the faults of the original. The biggest difference is that an additional keypad was added so that it is not necessary to toggle the Num Lock key to alternate between using the keypad to type numbers and move the cursor. The Return key was also enlarged, the Backslash key was moved to the far side of the Right Shift key, and the Grave key ( ) was moved above the Return key, to leave more room for the latter.

In addition, the function keys were moved from the dual row on the left side of the keyboard to a single row above the main portion of the keyboard. Indicator lights were added to the toggle keys, and an Enter key (which is equivalent to the Return key) was added to the existing numeric keypad.

The feel of the keys is mushy, and, although you're unlikely to press the backslash key accidentally with the 5151's layout, it's also hard to get to it (without looking) when you do want to press it. In addition, KeyTronic placed the Caps Lock key between the "A" key and the Ctrl key, so it's easy to hit it by mistake.

A Zenith clone

The keyboard Zenith Data Systems sells with several computers is shown in Fig. 3. It corrects many of the faults of the original IBM board, has a slightly mushy feel (but not as much as the KeyTronic model), and emits a nice keyclick (through the speaker in the system unit). The Return key is large, and there is an Enter key in the numeric keypad. The Backslash key was moved down a row, so that it's between the Spacebar and the Alt key. You're not likely to hit it accidentally, but it can be hard to home in on, especially if you also use an IBM or other keyboard. One nice feature is that the toggle keys all have internal LED's. The keyboard is available separately; see Table 1 for more detailed information.

The AT layout

Everyone in the industry knew that there was widespread dissatisfaction with the original IBM layout, but IBM ignored that dissatisfaction when it introduced the XT in 1983. In fact, it wasn't until the AT was introduced in 1984 that IBM attempted to correct its error. We were unable to obtain an AT keyboard to photograph, but the layout of the AT-style clone keyboard (shown in Fig. 4) is quite similar. Notice that the Enter key is much larger now. However, the BackSpace key is smaller, and the Backslash key has been moved to the upper row (between the BackSpace and the "=" keys).

FIG. 1—IBM'S ORIGINAL KEYBOARD

FIG. 2—THE KEY TRONIC KB5151, AN EARLY CLONE

FIG. 3—A ZENITH CLONE

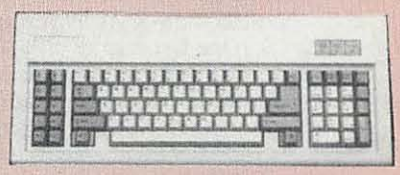


FIG. 4—A FOREIGN CLONE (IN THE AT STYLE)

The clone board shown in the photo has an Enter key in the keypad, but the AT does not. In addition, on the AT, the upper right keypad key is labeled SysReq; that key is not normally used by DOS. Beneath it is the PrtSc key, followed by the "-" key and the "+" keys. On the AT, the "+" key occupies the space of two keys.

The strangest thing about the AT layout is that the Esc key has been moved from its traditional place at the upper left corner of the main keyboard to the upper left corner of the numeric keypad. One can only speculate why that was done. The Grave key was moved to the normal Esc position. The AT keyboard also has indicator lights for the toggle keys.

IBM's Enhanced keyboard

Two years after the announcement of the AT, IBM introduced the XT 286, a machine

TABLE 1—PRODUCTS REVIEWED

Product	Manufacturers
RapidWriter $595	Quixote Corporation One East Wacker Drive Chicago, IL 60601. 800-325-1850 (Illinois) 800-523-8356 (Elsewhere)
KB5151 $255 **KB5153** $299	Key Tronic Corporation P.O. Box 14687 Spokane, WA 99214 800-262-6006
Turbo-101 $149.95	DataDesk International 7650 Haskell Avenue Van Nuys, CA 91406 800-826-5398
100-1861 $125	Heath Company Parts Department Benton Harbor, MI 49022 616-982-3571

Contact your local IBM representative for information on the IBM Enhanced keyboard.

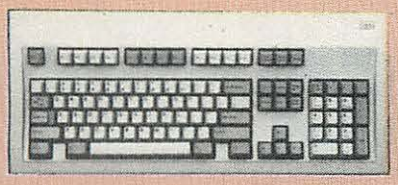


FIG. 5—IBM'S ENHANCED KEYBOARD

FIG. 6—THE DATADESK TURBO-101

FIG. 7—QUIXOTE CORPORATION'S RAPIDWRITER

FIG. 8—THE KEY TRONIC KB5153

halfway between the XT and the AT in computing power. Along with the XT 286 came a new keyboard, the so-called Enhanced keyboard. If you had never seen a computer keyboard before, you'd probably say that it's a work of art. (See Fig. 5.) The main keyboard has a symmetrical layout and there are separate numeric and cursor keypads. Also, there are two new function keys, for a total of twelve function keys, and 101 keys overall.

The feel of the new keyboard is wonderful; you don't have to press the keys very far or very hard. There's no audio feedback, but you don't need it; the keys themselves provide a pleasant yet unobtrusive click.

In fact, there's only one real problem with the Enhanced keyboard: the Ctrl key. Actually there are two Ctrl keys and two Alt keys, located symmetrically on both sides of the Spacebar. The normal position for the Ctrl key is now occupied by the Caps Lock key. If you're used to a keyboard with Ctrl in the normal position, you may go wild trying to adapt to the new position.

Another anomaly of the Enhanced keyboard is that the Esc has been moved yet again—now it's at the left edge of the upper row of keys, by the Function keys. At least that's the same general area as normal.

The numeric keypad finally has an Enter key, and the Backslash key is now in a reasonable location: just above the Enter key in the main keyboard. In addition, the Back-Space key is now large and easy to find; it's located just above the Backslash key.

The DataDesk Turbo-101

There's always someone waiting to correct an IBM mistake. The Turbo-101 has much going for it, including a switch that allows you to swap the functions of the Caps Lock and the Left Ctrl keys, and keycaps to make it look as if the keyboard had been designed that way from the beginning.

In addition, another switch adapts the board for use with either a PC (or XT) or an AT. The board can also be used with either the old BIOS ROM (which doesn't recognize the new function keys) or new BIOS ROM's (which do). If you use the Turbo-101 with an old BIOS ROM, F11 generates Alt-F9 and F12 generates Alt-F10 key codes. By cutting the leads of two diodes, you can force the keyboard to generate the new scan codes for those keys. Further, the Turbo-101 comes with a copy of Borland's Turbo Lightning, a combination spelling checker and thesaurus.

The only problem with the keyboard is that each key has a soft detent that is inferior to that of IBM's Enhanced keyboard. Other than that, the Turbo-101 is a good deal.

Special keyboards

Several firms have taken the idea of improving the IBM keyboard further than merely re-arranging the layout. For example, the RapidWriter (shown in Fig. 7) is a hardware/software combination that is designed to increase secretarial efficiency by automating the process of typing repetitive words and phrases.

The keyboard is identical to the KeyTronic 5151. (Just as we were going to press Quixote Corporation informed us that future versions of RapidWriter will come with a 101-key enhanced keyboard. The software has also been upgraded.)

The software loads a special keyboard driver that senses when several keys are pressed simultaneously. That condition is called a "chord" by Quixote. When the keys corresponding to a previously stored chord are pressed, an entire word or phrase flows into the current document, just as if that word or phrase had been typed at the keyboard. Chords are stored in dictionaries; each dictionary can contain 250 chords for a total of 16,000 characters. You can have an unlimited number of dictionaries, as each is stored in a separate disk file.

You can cause the first letter of a chord to be capitalized by pressing a Shift key when you press the chord. Or you can capitalize the entire chord by pressing Caps Lock with the chord. In addition, you can define chords that pause one or more times during chord expansion, allowing you to type information at the keyboard. And a chord can "call" another chord, expand it, and return to the calling chord. You can also edit and print chords.

KeyTronic 5153

The most innovative and useful special keyboard we've seen is the KeyTronic 5153 (shown in Fig. 8), because it contains a programmable keypad (on the right side of the unit) instead of separate numeric and cursor keypads.

The basic keyboard layout is in the AT style, with the Escape key in the numeric keypad. The keyboard has the typical KeyTronic feel—mushy, but not so mushy as some inexpensive clones. The programmable keypad is like a digitizing tablet; it can resolve motion to a precision of about 0.001 inch. You use the keypad by pressing it with your finger or with a plastic stylus.

The keypad has several modes of operation. You can use it as a cursor keypad, in which each press is converted into equivalent cursor-key codes. In the function-key mode, the pad is divided into a number of squares, each of which is freely programmable. KeyTronic supplies program files (and plastic overlays) for common DOS commands (DATE, TIME, TYPE, FORMAT, etc.), and for popular applications programs, including WordStar and Lotus 1-2-3. KeyTronic also supplies software that allows you to create your own keypad macro files (for matrix sizes of 2×2, 3×3, 4×4, or 5×5), and blank overlays.

In the mouse mode, the keypad emulates operation of the Microsoft mouse; to use it, you run the stylus across the keypad. A graphics mode functions similarly, but each point on the keypad corresponds to a point on the display screen. You can also use the keypad in several modes that combine the above modes.

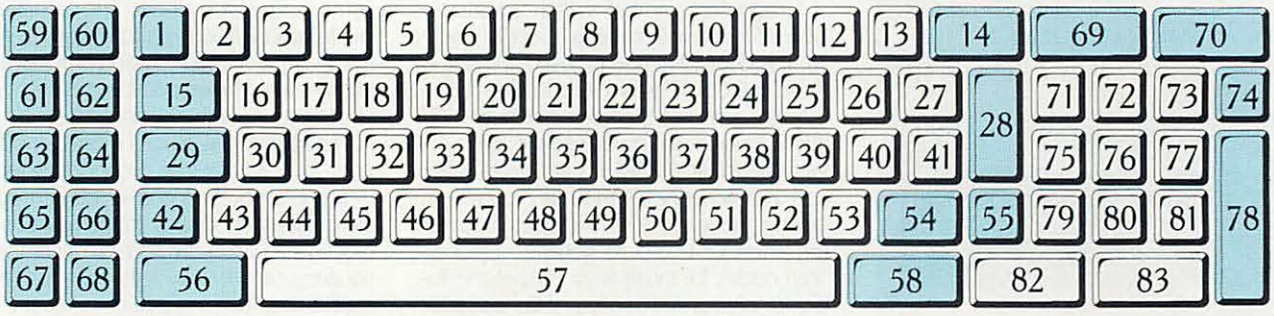
Conclusions

Each of the keyboards we examined has merit; some are better for particular applications than others. The main features you'll want to consider when buying a keyboard are the overall layout, indicator lights, feel (mushy, "clicky," or somewhere between the two), and extra features (bundled software, for example). The most important feature is layout, so examine it carefully; layout can make the difference between productive and non-productive use of a machine. Try before you buy.

Which would we choose? For general use, the IBM Enhanced keyboard. It has by far the best feel, and is now the standard of IBM's entire line of PC's.

The KeyTronic 5153 is our runner-up. It has a good feel, and the keypad can save the cost of a mouse, a digitizing tablet, or both. A person just starting out in computing could get by with it until he or she could justify the cost of the extra peripheral.

FROM KEYPRESS TO SCAN CODE



IBM keyboards come in a variety of sizes and shapes—here's how they work and how they differ from one another.

JEFF HOLTZMAN,

TECHNICAL EDITOR

IBM's *Technical Reference* manuals are notorious sources of both information and misinformation—or perhaps we should say non-information. Keyboard documentation is a perfect example. The "schematic diagram" (from the manual for the original PC) is little more than a block diagram, and is little help in understanding how the keyboard works. The circuit functions basically as discussed in our other article on keyboards this issue, except that the 8048 microprocessor performs the matrix scanning that the logic IC's do in the discrete-logic versions.

The most informative (and interesting) information about how the keyboard functions in the IBM PC is contained in the software listings in IBM's *Technical Reference* manuals. In this article we'll discuss how the software processes the raw key codes generated by the hardware, and we'll present a BASIC program that demonstrates graphically how your keystrokes are interpreted at various levels by the computer.

Hardware, BIOS, DOS

You can view the IBM PC from three different perspectives, as shown in Fig. 1. At the lowest level is the hardware: the microprocessor, RAM and ROM memory, the disk drives, the display adapter card, the monitor—and the keyboard. Controlling the hardware, of course, is software; and, in the IBM, there are two levels of control software.

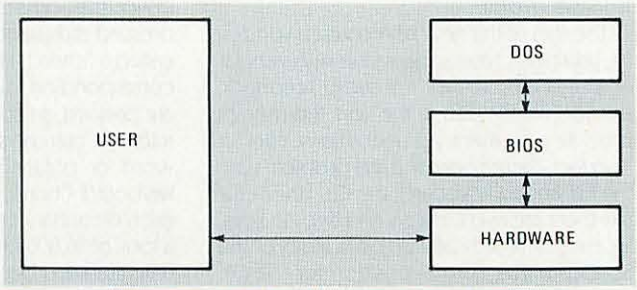


FIG. 1—HARDWARE AND TWO LEVELS OF SOFTWARE comprise the IBM PC. The user interacts with the hardware, which communicates with the BIOS. DOS provides a number of high-level functions for programmers to use in accomplishing a wide variety of tasks, including reading and writing disk drives, communications ports, and the keyboard.

The lower of those two levels is the BIOS (Basic Input/Output System); it is contained in a ROM (or in an EPROM on most clones). Code in the BIOS ROM is responsible for all of the low-level functions of the computer: displaying characters on the screen, sending them to the printer, transferring data to and from the disk drives, getting keystrokes from the keyboard, etc.

The upper software level is the DOS (Disk Operating System), which is contained in several files on disk. DOS is the level through which programmers are *supposed* to display data, manipulate disk files, and get keystrokes. However, to improve performance (or sometimes just by preference), many programmers go to the BIOS (or even to the hardware itself).

Interrupts

To understand how the hardware communicates with the software, you must understand the basics of interrupts. A device (the keyboard, for example) can interrupt the normal processing of the computer's 8088 microprocessor. When you press a key at the keyboard, it generates a signal that is sent to the computer; that signal says "Hey! Somebody pressed a key!"

The microprocessor then stops what it is doing and loads the address corresponding to the keyboard handler (interrupt 9) from a special location in memory. (Unless otherwise specified, all numbers in this article are in decimal notation). Processing continues at that address as the 8088 reads the keyboard port, converts the raw key code into something meaningful, and stores it for use by whatever program was running before the interrupt occurred. Last, the 8088 performs a special instruction (IRET, for Interrupt Return) that allows it to continue where it left off before the interrupt took place.

Devices other than the keyboard (the disk drives and the serial ports, for example) generate their own interrupts, which the 8088 processes in the same fashion. The difference is that each interrupt is directed to a different location in memory.

In addition to hardware interrupts, the 8088 also allows software interrupts for many commonly used functions. For example, when you press Shift-PrtSc, whatever is displayed on the screen is sent to the printer. That works as follows: First the two keypresses (Shift and PrtSc) generate their own interrupts. The computer processes those interrupts one at a time, and, when it realizes that a print-screen operation should be performed, it generates interrupt 5 (from within the interrupt-9 handler). Interrupt 5 does the screen-print and then returns to the interrupt-9 handler, which then returns to whatever program was in control when Shift-PrtSc was pressed.

Other software interrupts, at both BIOS and DOS levels, allow many operations, including displaying characters, getting user input from the keyboard, reading and writing disk drives, reading and writing communications ports, etc. As we saw, interrupts can interrupt each other (sometimes—but that's a story that we'll not get into here.) With the basics of interrupts in mind, now let's see how the keyboard-interrupt-processing software works.

Keyboard hardware, BIOS software

Keyboard processing on the IBM provides a good example of how the hardware meshes with the software. As we said, each keypress generates an interrupt 9. That interrupt is processed in the BIOS ROM and then passed on to DOS for further, more sophisticated handling.

There are 83 keys on the standard IBM keyboard; each has an associated eight-bit *scan code*. The keys and their scan codes are shown in our lead illustration and in Table 1. Each time you press a key (any key, including the ones you don't normally think of as generating a code—the Shift keys, Alt, Ctrl, etc.), the keyboard interrupts the microprocessor, sending it the scan code. Each time you release a key, the keyboard generates another interrupt, sending it the same scan code, but now with the high bit set (i. e., the scan code + 128).

TABLE 1—IBM KEYBOARD CODES

Code	Label	Code	Label	Code	Label	Code	Label
1	Esc	22	U	43	\	64	F6
2	1	23	I	44	Z	65	F7
3	2	24	O	45	X	66	F8
4	3	25	P	46	C	67	F9
5	4	26	[	47	V	68	F10
6	5	27	]	48	B	69	NumLk
7	6	28	Retn	49	N	70	ScrLk
8	7	29	Ctrl	50	M	71	Home
9	8	30	A	51	,	72	UpArw
10	9	31	S	52	.	73	PgUp
11	0	32	D	53	/	74	—
12	—	33	F	54	RShft	75	LftAr
13	=	34	G	55	PrtSc	76	5
14	Bksp	35	H	56	Alt	77	RgtAr
15	Tab	36	J	57	Space	78	-
16	Q	37	K	58	CapLk	79	End
17	W	38	L	59	F1	80	DwnAr
18	E	39	;	60	F2	81	PgDn
19	R	40	/	61	F3	82	Ins
20	T	41	\	62	F4	83	Del
21	Y	42	LShft	63	F5		

The BIOS then translates the scan codes into ASCII and other codes, depending on the state of eight keys: Control, Alt, Delete, Insert, Left Shift, Right Shift, Num Lock, Caps Lock, and Scroll Lock. For example, the "A" key has a (hardware-level) scan code of 30. So when that key is pressed, the 8048 in the keyboard sends a 30 to the IBM BIOS through Interrupt 9. When the key is released, the keyboard sends a 158 (30+128) to the computer. If the "A" key is pressed continuously, the 8048 continuously sends 30's until the key is released, at which time a 158 is sent.

The BIOS would translate that 30 into a lowercase "a" (ASCII 97). But suppose that one of the shift keys were pressed simultaneously with the "A." In that case, the BIOS would translate that 30 into an uppercase "A" (ASCII 65). If the Control key were pressed, the hardware-level 30 would become a BIOS-level Cntl-A (ASCII 1). If Caps Lock were on and one of the shift keys were pressed, a lowercase "a" would be generated.

However, if the Alt key is pressed with the "A" key, something funny happens: The BIOS now generates two codes, the first of which is a zero, and the second of which is often (but not always) the scan code for that key. And the scan code, of course, bears no relation to standard ASCII codes. The Function keys, the arrow keys,

LISTING 1

```
10 REM Computer Digest Keyboard Demo
20 REM Copyright        Jeff Holtzman          03-27-87
30 GOSUB 2000
100 REM main loop ------------------------------------------------
110 GOSUB 150                       : REM update shift key display
120 A$=INKEY$
130 IF A$<>"" THEN GOSUB 200        : REM update DOS display
140 GOTO 110
150 REM Update status key display --------------------------------
160 YBOX=21:XBOX=-7:FLAG1=PEEK(&H17)
170 FOR SK=1 TO 8
175     XBOX=XBOX+10
180     IF FLAG1 AND 2^(SK-1) THEN GOSUB 500 ELSE GOSUB 400
185 NEXT
190 RETURN
200 REM Display DOS keys -----------------------------------------
205 A=ASC(A$)
210 LOCATE 13,36:PRINT A;
220 IF A=0 THEN PRINT ASC(MID$(A$,2)); ELSE PRINT "   ";
230 LOCATE 5,22
235 IF INSTR(NOPRINT$,A$)=0 THEN PRINT A$; ELSE PRINT " ";
240 LOCATE 5,32:PRINT "                          ";
250 LOCATE 5,32
260 IF A > 127 THEN PRINT "Extended (8-bit) ASCII";:GOTO 300
270 IF A > 31 THEN PRINT "Regular (7-bit) ASCII";:GOTO 300
280 IF A > 0 THEN PRINT "Cntl-";CHR$(A OR &H40);:GOTO 300
290 PRINT DESC$(ASC(MID$(A$,2)));
300 RETURN
400 REM Clear box ------------------------------------------------
410 REM inputs: xbox,ybox
420 LOCATE YBOX,XBOX
430 PRINT "      ";
440 LOCATE YBOX+1,XBOX
450 PRINT "      ";
460 RETURN
500 REM Fill box -------------------------------------------------
510 REM inputs: xbox, ybox
520 LOCATE YBOX,XBOX
530 PRINT FILL$
540 LOCATE YBOX+1,XBOX
550 PRINT FILL$
560 RETURN
600 REM draw box -------------------------------------------------
610 REM inputs: xbox,ybox : start position
630 REM         xlen,ylen : length
640 REM first do corners
650 LOCATE YBOX,XBOX
660 PRINT CHR$(201);
670 LOCATE YBOX,XBOX+XLEN-1
680 PRINT CHR$(187);
690 LOCATE YBOX+YLEN-1,XBOX
700 PRINT CHR$(200);
710 LOCATE YBOX+YLEN-1,XBOX+XLEN-1
720 PRINT CHR$(188);
730 REM now do horiz lines
740 FOR BOX=1 TO XLEN-2
750    LOCATE YBOX,XBOX+BOX
760    PRINT CHR$(205);
770    LOCATE YBOX+YLEN-1,XBOX+BOX
780    PRINT CHR$(205);
790 NEXT BOX
800 REM now do vert lines
810 FOR BOX=1 TO YLEN-2
820     LOCATE YBOX+BOX,XBOX
830     PRINT CHR$(186);
840     LOCATE YBOX+BOX,XBOX+XLEN-1
850     PRINT CHR$(186);
860 NEXT BOX
870 RETURN
900 REM Draw shift keys ------------------------------------------
910 YBOX=20:XLEN=8:YLEN=4
930 FOR FKEY=2 TO 72 STEP 10
940    XBOX=FKEY
950    GOSUB 600
960 NEXT
970 RESTORE 1070
980 FOR FKEY=3 TO 73 STEP 10
990     LOCATE 18,FKEY
1000    READ A$
1010    PRINT A$
1020    LOCATE 19,FKEY
1030    READ A$
1040    PRINT A$
1050 NEXT
1060 RETURN
1070 DATA Right,Shift,Left,Shift,,Control,,Alt
1080 DATA Scroll,Lock,Num,Lock,Caps,Lock,,Insert
1100 REM Draw BIOS box -------------------------------------------
1110 LOCATE 11,30
1120 PRINT "BIOS-level scan code ";
1130 XBOX=35:YBOX=12:XLEN=10:YLEN=3
1140 GOSUB 600
1150 RETURN
1200 REM Draw DOS boxes ------------------------------------------
1210 XBOX=20:YBOX=4:XLEN=5:YLEN=3
1220 GOSUB 600
1230 LOCATE 7,20
1240 PRINT "ASCII";
1250 XBOX=30:YBOX=4:XLEN=26:YLEN=3
1260 GOSUB 600
1270 LOCATE 7,32
1280 PRINT "Description";
1290 LOCATE 3,31
1300 PRINT "The DOS Interpretation";
1310 RETURN
2000 REM Initialize ----------------------------------------------
2010 KEY OFF:CLS:DEFINT A-Z:DEF SEG=&H40
2020 FOR I=1 TO 6:FILL$=FILL$+CHR$(219):NEXT
2030 FOR I=1 TO 10:KEY I,"":NEXT
2040 NOPRINT$=CHR$(7)+CHR$(9)+CHR$(10)+CHR$(12)+CHR$(13)
2050 LOCATE 1,19,0
2055 PRINT "Computer Digest Keyboard Demonstration";
2060 GOSUB 900  : REM draw shift keys
2070 GOSUB 1100 : REM draw BIOS box
2080 GOSUB 1200 : REM draw DOS box
2090 DIM DESC$(132)
2100 RESTORE 3010
2110 FOR I=1 TO 132:READ DESC$(I):NEXT
2120 RETURN
3000 REM Key label data ------------------------------------------
3010 DATA ,,Null,,,,,,,,,,
3020 DATA Back-Tab
3030 DATA Alt-Q,Alt-W,Alt-E,Alt-R,Alt-T,Alt-Y,Alt-U,Alt-I,Alt-O,Alt-P
3040 DATA ,,,
3050 DATA Alt-A,Alt-S,Alt-D,Alt-F,Alt-G,Alt-H,Alt-J,Alt-K,Alt-L
3060 DATA ,,,,
3070 DATA Alt-Z,Alt-X,Alt-C,Alt-V,Alt-B,Alt-N,Alt-M
3080 DATA ,,,,,,,
3090 DATA F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,,
3100 DATA Home,Up-arrow,PgUp,,Left-arrow,,Right-arrow,,End,Down-arrow
3110 DATA PgDn,Ins,Del
3120 DATA Shift-F1,Shift-F2,Shift-F3,Shift-F4,Shift-F5
3130 DATA Shift-F6,Shift-F7,Shift-F8,Shift-F9,Shift-F10
3140 DATA Cntl-F1,Cntl-F2,Cntl-F3,Cntl-F4,Cntl-F5
3150 DATA Cntl-F6,Cntl-F7,Cntl-F8,Cntl-F9,Cntl-F10
3160 DATA Alt-F1,Alt-F2,Alt-F3,Alt-F4,Alt-F5
3170 DATA Alt-F6,Alt-F7,Alt-F8,Alt-F9,Alt-F10
3180 DATA Cntl-PrtSc,Cntl-Left-arrow,Cntl-Right-arrow
3190 DATA Cntl-End,Cntl-PgDn,Cntl-Home
3200 DATA Alt-1,Alt-2,Alt-3,Alt-4,Alt-5
3210 DATA Alt-6,Alt-7,Alt-8,Alt-9,Alt-0
3220 DATA Alt-,,Alt=,Cntl-PgUp
```

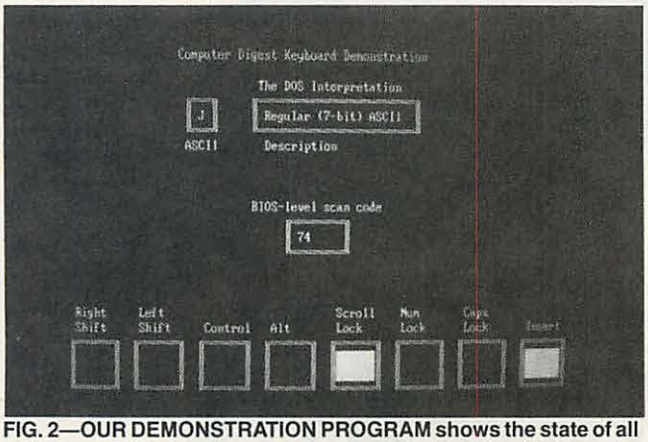


FIG. 2—OUR DEMONSTRATION PROGRAM shows the state of all shift keys, BIOS-level scan codes, and the high-level interpretation of those codes.

Home, etc., all generate the two-byte codes.

If you want to see which codes are generated by which keys (and combinations of keys), the program in Listing 1 provides a graphic representation of how those keys are interpreted. You can download the program (KEYMON.BAS) from our BBS (516-293-2283); if you type it in yourself, make sure you enter all the commas in the DATA statements.

For example, after pressing Scroll Lock, Insert, and the "J" key, the screen appears as shown in Fig. 2. You'll notice that the boxes corresponding to the Shift keys, Ctrl, and Alt light up as long as you press those keys and go dim when you release them. By contrast, Num Lock, Scroll Lock, Caps Lock, and Insert are toggles—each time you press one of those keys, an internal flag is alternately set and reset that indicates the given state (on or off).

You'll notice that some keys and key combinations produce no display. For example, the "5" key in the keypad produces no code when Num Lock is off. It's important to understand that every time you press *any* key the keyboard generates an interrupt (unless the keyboard buffer is full, at which point the keyboard will beep). If pressing a key produces no apparent result, that's because the BIOS has defined no code for that key (or combination).

Some programs make use of the "undefined" keys. For example, Cruise Control (reviewed in this month's Editor's Workbench), uses that "5" key (when Scroll Lock is off) as a special hotkey for controlling various functions. However, to get at those undefined keys, you have to write a complete Interrupt 9 handler—and that's no trivial pursuit.

Our demonstration program has several "bugs." Those bugs are due to differences between the ways that BASIC and DOS treat the keyboard. For example, if you print a CHR$(12) to the screen in BASIC, the screen will be cleared. In DOS, however, you'll see the "female" symbol (a circle over a cross). There are several such anomalies; tracking them down will teach you much about BASIC and DOS, as well as the keyboard.

ASCII, extended ASCII, and special codes

Basically, ASCII is a seven-bit code that provides a total of 128 (2^7) unique codes. However, personal-computer memory is organized in eight-bit (or 16-bit) chunks. So why didn't IBM encode all the special keys in the upper 128 ASCII codes? The reason is that IBM wanted to retain the upper codes for use by displayable characters. For example, most of the codes from 128 to 167 are foreign-language characters. Others include box-drawing characters, special math symbols, etc.

You know how to type in standard ASCII codes and the two-byte special codes—but how do you type in the extended ASCII codes? Some programs let you do so directly (for example, by associating special characters with the Function keys); the IBM BIOS lets you type in *any* ASCII code from 1 to 255 as follows. Press the Alt key, and hold it down. Now type the three-digit decimal code that corresponds to the desired character. Use only the keypad keys, not the number keys above the main keyboard. After you release the Alt key, the character will be displayed. That procedure works in BASIC, at the DOS command line, and in some (but not all) applications programs.

New keyboards

When IBM introducted the IBM PC AT in 1984, it introduced a new keyboard. The AT keyboard has a new layout (as shown in the review in Editor's Workbench this issue), and it works differently. The biggest hardware difference is that the keyboard now both transmits and receives data. You can force it to stop scanning temporarily, resume scanning, set the "Typematic" (repeat) rate, and turn the status-indicator LED's on and off.

In addition, the hardware-level scan codes have changed. The keys have different numbers, and there is one new key. However, those hardware differences are transparent at all levels above (and including) the BIOS. So our demonstration BASIC program works on the AT. But any program that works with the keyboard at the Interrupt 9 level must know whether it is running on an AT or a standard PC.

IBM still wasn't satisfied with the state of keyboard confusion, so, when the company introduced the XT 286 last fall, it introduced yet another keyboard. The new keyboard has 101 keys, even more commands issuable by the system, and three (!) software-selectable sets of scan codes. The first set is similar to the PC/XT set; the second set is similar to the AT set; and the third set is similar to the AT set, except that every key generates a unique code, regardless of the state of any of the shift keys (including Cntl, Alt, etc.) The last set should make it unnecessary for keyboard-enhancement programs to take over the keyboard-processing interrupts completely. However, such programs will still have to contend with the PC/XT and AT keyboards. Set 2 is the power-up default set.

To give you some idea of how the three sets of scan codes are related, consider this example. The enhanced keyboard has two Insert keys, one in the numeric pad (key 99), the other in the new cursor-control pad (key 75) located between the typewriter and the numeric-keypad sections. Table 2 shows the codes that are generated from each set when Insert is pressed and no shift keys are pressed.

TABLE 2—ENHANCED KEYBOARD INSERT CODES

Code	Key 75		Key 99	
Set	Make	Break	Make	Break
1	E0 52	E0 D2	52	D2
2	E0 70	E0 F0 70	70	F0 70
3	67	F0 67	70	F0 70

Programming and the special keys

In BASIC, you can use the normal INKEY$ function to get both standard and special keys. Normally INKEY$ collects single characters, but when a special key is pressed, INKEY$ returns two characters, the first of which is a CHR$(0). As at the BIOS level, that's a sign that another character is available. You can test for the existence of a special code by checking the length of the string that INKEY$ returns. Our demonstration program illustrates the procedure; see lines 100–220.

If you're interested in working with the keyboard in assembly language, you'll want to understand how BIOS interrupts 9 and 16, and DOS interrupt 33 (function calls less than 10) work. The best sources of information for BIOS listings and scan codes are IBM's *Technical Reference* manuals for the PC (or the XT), the AT, and the XT 286, as well as the DOS *Technical Reference* manual. Some of those manuals are hard to obtain (and expensive), so you may wish to consult Peter Norton's *Programmer's Guide To The IBM PC* and Ray Duncan's *Advanced MS DOS*. Both are published by Microsoft Press, and both are excellent sources of information on the BIOS, DOS and other subjects.

WORKING WITH SURPLUS KEYBOARDS

How they work, and how to use 'em.

Robert Grossblatt

Once upon a time, most of us had to throw switches and turn dials, but if you spent some extra bucks, you could talk to your equipment by pressing buttons. Things stayed like that until calculators showed up. When computers hit the market, keyboards became commonplace.

Adding a keyboard to your own circuit is easy. And with the parts market loaded with surplus keyboards, it's inexpensive. But using a keyboard successfully means understanding how it works, how it's driven, and what you need to get it working. Once we know the theory, we'll talk about how to use those surplus keyboards sold in the back of this magazine.

All keyboard circuits are made up of three parts—the switches themselves, decoding circuitry, and encoding circuitry. The keys are wired so that each one produces a unique code that can be passed on to the decoder, the circuit's main section. The encoder will take the keypress and translate it into whatever kind of information is needed by the equipment the keyboard is talking to. Let's discuss each circuit in turn.

Two methods

The two methods most frequently used to wire up switches are with a common leg and in a row-and-column matrix. In Figure 1 you can see that both arrangements will let each keypress generate a unique code. A common-leg set-up (Fig. 1-a) is much simpler to design but is only suited to applications where a few switches are needed. Since each switch you add means another lead coming from the keyboard, large numbers of switches become wiring nightmares. A matrix keyboard (Fig. 1-b) has fewer connections but it usually needs more support circuitry.

The break-even point for connections is eight switches. A common-leg keyboard that size will need nine leads, and a matrix keyboard will need eight. Since there are advantages and disadvantages to both, which is best depends on what you're doing.

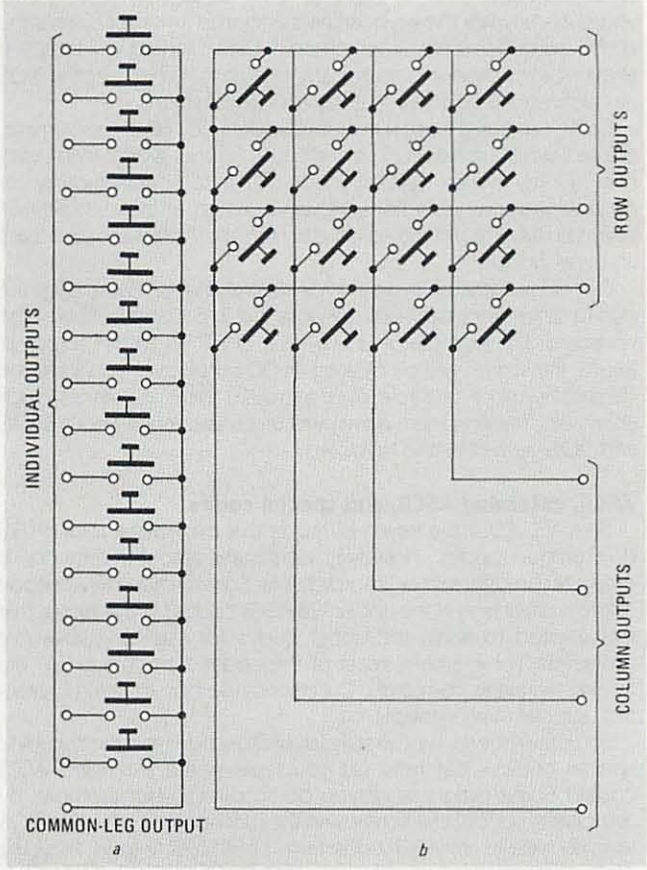


FIG. 1—A KEYBOARD can be wired from a linear array of switches (*a*) or in an X-Y matrix (*b*).

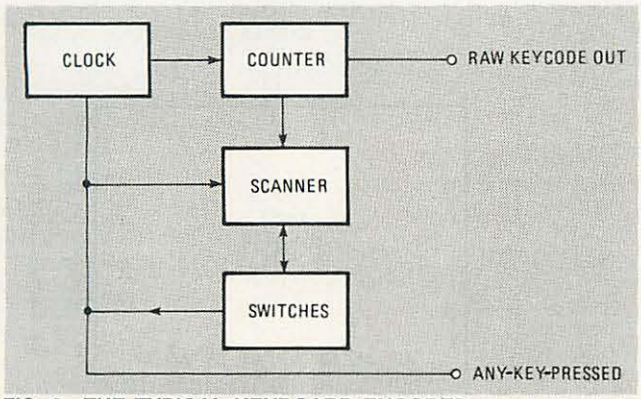


FIG. 2—THE TYPICAL KEYBOARD ENCODER uses a clock to increment a counter that controls matrix scanning.

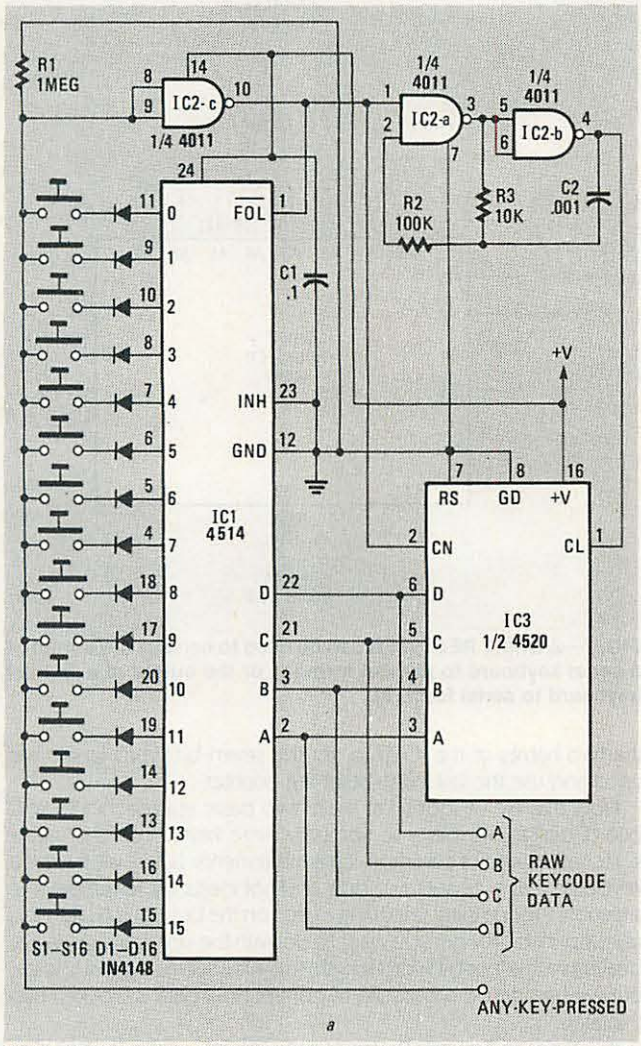


FIG.3—A KEYBOARD ENCODER. The scanning circuitry for a common-leg keyboard is shown here.

However you wire your switches, the signals they generate must be decoded. The circuit must recognize that a key is being pressed, figure out which one it is, and then put the appropriate code elsewhere. Decoders can be put together with anything from a handful of resistors and diodes to a microprocessor and a little bit of digital glue.

In all but the simplest keyboards, the decoder uses some sort of clock to scan the switches looking for a keypress. Figure 2 is a block diagram of this kind of circuit. The output of the counter makes the scanner sample each switch on the keyboard. When a key is pressed, the clock is stopped, the count is frozen, and the Any-Key-Pressed line becomes active. That signal tells some other circuitry down the line that the keyboard is putting out data.

That circuit could be used for both matrix and common-leg keyboards. The difference between the two would be in how the keyboard was scanned. Figure 3 shows the scanning circuitry for a common-leg keyboard, and Fig. 4 shows a similar setup using a matrix keyboard. The clock and counting circuitry is the same.

In Fig. 3, the values of R2, R3, and C2 give the clock composed of IC2-b and IC2-c an output frequency of about 100 kHz. That signal drives both IC3, half a 4520 binary counter, and IC1, a 4514 1-of-16 line decoder. As the count cycles from 0 (0000) to F (1111), each of IC1's outputs goes high in turn. R1 serves two purposes—it holds the common leg of the switches low, and, with C1, helps to debounce the switches.

When a key is pressed, nothing happens until that output of the 4514 is selected by the count of the 4520. When the output does go high, the Any-Key-Pressed line goes high, IC2-a inverts the signal and disables the clock and the counter, and puts a low on pin 1 of the 4514 to disable it also. The result of this is that a keypress freezes the output data lines at the selected number and generates a signal to indicate that valid data is on the bus.

There are two features of the circuit that should be noticed. First, although the switches are debounced, the design of the keyboard eliminates switch bounce. If you used noisy switches, the worst that would happen is that the switch would be in an open con-

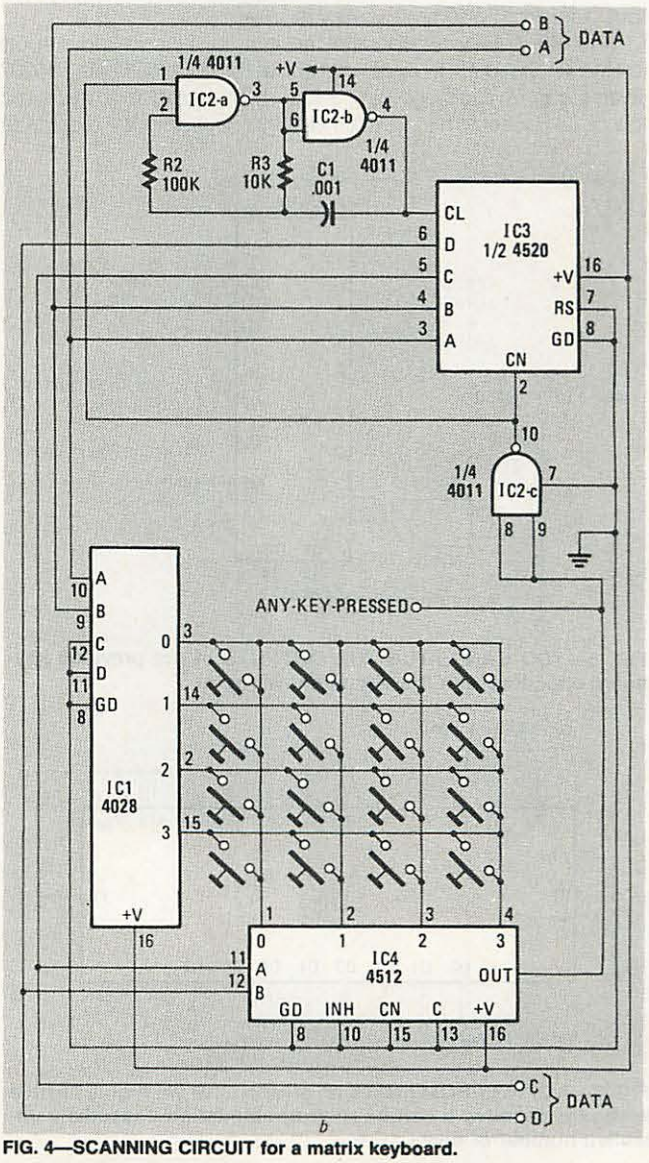


FIG. 4—SCANNING CIRCUIT for a matrix keyboard.

dition when its output was selected. In that case, the circuit would cycle through another count as the clock kept running. Only a valid keypress would produce valid data.

Two-key rollover

That circuit has two-key rollover. D1 to D16 isolate each of the 4514's output lines, so, if two keys are closed at the same time, the circuit will output the second bit of data as soon as the first key is released.

Figure 4 shows a circuit for a matrix keyboard. Though we're encoding the same number of switches as we did in the common leg arrangement, we only need nine leads from the keyboard instead of seventeen. The setup is different too. The basic idea behind using a matrix keyboard is to have the control signal come in on one side of the matrix and leave on the other. In Fig. 4, IC4 is a 4028 BCD-to-decimal converter. A binary address on the inputs causes the selected output to go high while all the rest remain low. As we're only handling a four-by-four switch matrix, we only need two of the inputs.

The two low-order bits from the 4520 are routed to the 4028, and the two high-order bits are routed to the 4512, an eight-channel data selector. When one of its inputs is selected, the signal at the input appears at the output. If a key is pressed, the high signal at the output of the 4028 is channeled through the 4512 and serves the same function as the common switch leg did in Fig. 3. It disables both the clock and the counter and also becomes our Any-Key-Pressed line to let other circuitry know that there's valid data on the bus. The circuit also has two-key rollover.

Look at Fig. 4—what about all those unused inputs and outputs on the 4028 and 4512? And what about the other half of the 4520? Even though we're only using a four-by-four keyboard, this same ci·cuit can be set to handle a ten-by-eight keyboard! We'd cascade

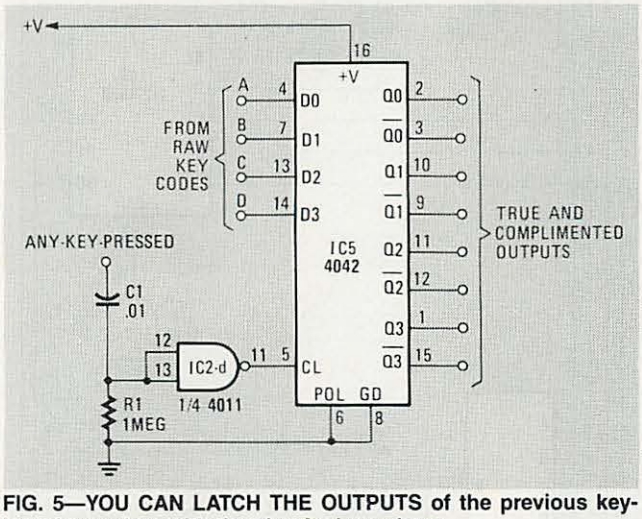


FIG. 5—YOU CAN LATCH THE OUTPUTS of the previous keyboard encoders with the circuit shown here.

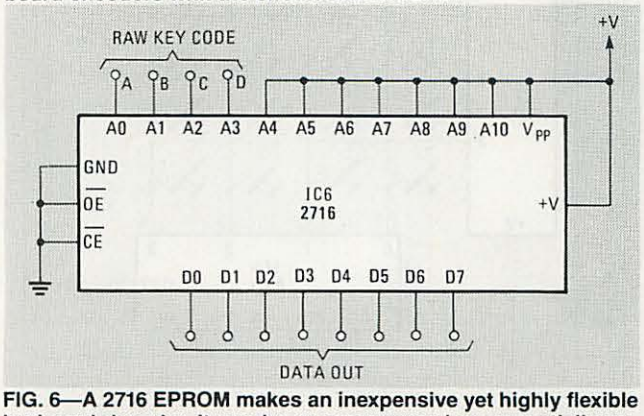


FIG. 6—A 2716 EPROM makes an inexpensive yet highly flexible keyboard decoder. It can be re-programmed an essentially unlimited number of times.

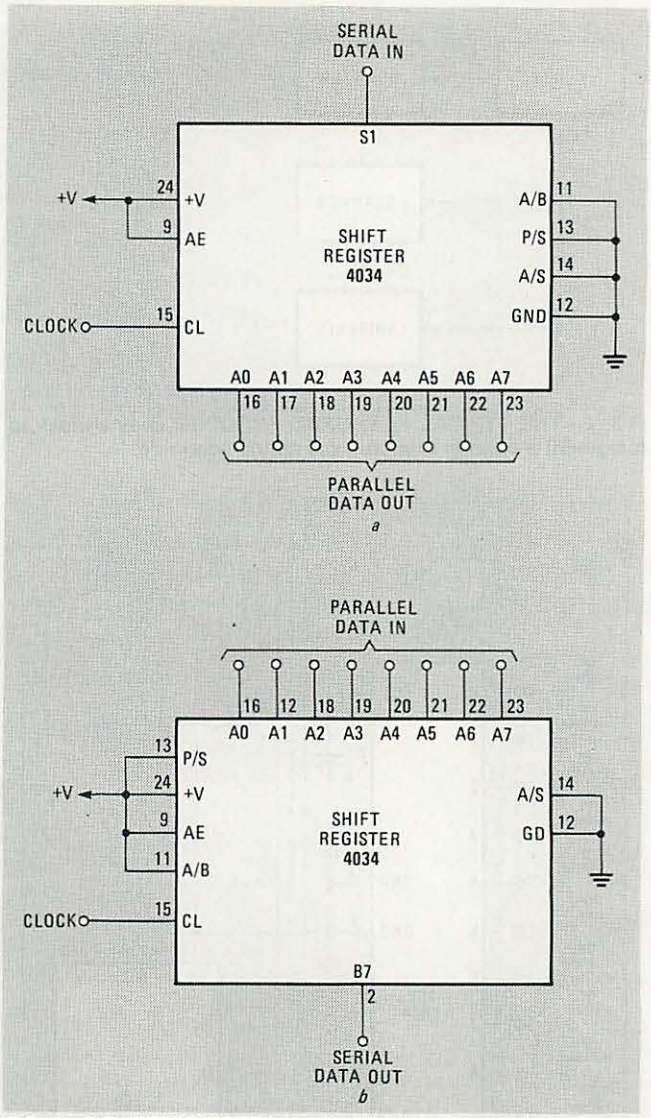


FIG. 7—A SHIFT REGISTER can be used to convert the output of a serial keyboard to parallel form (*a*), or the output of a parallel keyboard to serial form (*b*).

the two halves of the 4520 to get the seven-bit word length we need and use the last bit to reset the counter.

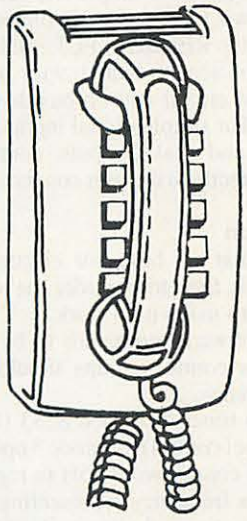
Now that we've looked at these two basic approaches to keyboard design, it's plain to see why large keyboards use matrix switches instead of common-leg arrangements. But all we have is a keyboard that puts out raw code, and not elegantly. To remedy the situation, the first thing is to hang a latch on the bus. Which latch you use depends on what you want to do with the keyboard. As we're dealing with a four-bit word length, the 4042 seems a good choice, but we have to do something to control how data is clocked into the latch.

If we use the Any-Key-Pressed line to directly control storing data in the latch, there's a chance we're going to get flaky behavior because of timing problems. Things have to happen in sequence. First valid data has to be on the bus, then it has to be clocked into the latch. The Any-Key-Pressed line has to signal something else that waits a while and then opens the latch for storage.

Since we have a NAND gate left over, we'll use it to build an edge detector and control the latch as shown in Fig. 5. Since the latch's polarity control, pin 6, is tied low, the latch will ignore its inputs as long as the store control, pin 5, stays high. Bringing the store input low will write data into the latch. The edge detector made from IC2-d will generate a negative-going pulse when it sees a positive pulse at its input. With the values given for C3 and R4, the pulse will be about 10 milliseconds wide.

R-E Engineering Admart

Rates: Ads are 2¼" × 2⅞". **One** insertion **$825. Six** insertions **$800** each. **Twelve** insertions **$775** each. Closing date same as regular rate card. Send order with remittance to **Engineering Admart,** Radio Electronics Magazine, 500-B Bi-County Blvd., Farmingdale, NY 11735. Direct telephone inquiries to Arline Fishman, area code-**516-293-3000.** ***Only 100% Engineering ads are accepted for this Admart.***



CALL NOW AND RESERVE YOUR SPACE

- 6 × rate $800.00 per each insertion.
- Reaches 239,312 readers.
- Fast reader service cycle.
- Short lead time for the placement of ads.

Call 516-293-3000 to reserve space. Ask for Arline Fishman. Limited number of pages available. Mail materials to: Engineering Admart, RADIO-ELECTRONICS, 500-B Bi-County Blvd., Farmingdale, NY 11735.

Circuit Board Layout Made Easy!

Create and Revise P-C-B Artwork on your

IBM or Compatible

- User Friendly
- Economical
- On-line Help Screen
- Supports Microsoft Mouse
- 2X artwork on printer

Requirements: IBM or compatible PC - 256K memory - CGA card - IBM graphics compatible printer.

ONLY $99.00

PCBoards

2145 Highland Ave./Ste. 201
Birmingham, Al. 35205
(205) 933-1122

DEMO DISK $10.00

CIRCLE 203 ON FREE INFORMATION CARD

117 PRACTICAL IC PROJECTS BUILD YOU CAN

2645T—117 PRACTICAL IC PROJECTS YOU CAN BUILD..... $10.95. Dozens of fully-tested, ready-to-build circuits you can put together from readily-available, low cost IC's! There are a total of 117 IC circuits ranging from an audio mixer and a signal splitter to a tape-deck amplifier and a top-octave generator organ! From TAB Books. **To order your copy send $10.95 plus $2.75 shipping to Electronic Technology Today Inc., P.O. Box 240, Massapequa Park, NY 11762-0240**

DIAGNOSE VHS VCR MECHANICS

WE'LL SHOW YOU HOW!

One of the most important VCR servicing tools for preventing tape damage is a precision torque gauge for Take-up, Stop, FF, and REW brake and clutch torque.

The New Tentel TQ-600 dial torque gauge comes complete with instructions and a specially modified cassette to allow critical readings to be made quickly and easily: Only **$139**; and there's a 30 day money back guarantee if you aren't 100% satisfied. Tentel also manufactures other test instruments to make VCR mechanical testing fast and easy. Call or send for details and ordering information today!.

Toll Free 800-538-6894
TENTEL Corp.
1506 Dell Ave
Campbell CA 95008

MasterCard VISA

408-379-1881
800-538-5894 (except Calif.)

CIRCLE 180 ON FREE INFORMATION CARD

LINEAR IC EQUIVALENTS & PIN CONNECTIONS

BP141—Shows equivalents & pin connections of a popular user-oriented selection of European, American and Japanese liner IC.'s 320 pages, 8 × 10 inches. **$12.50 Plus $2.75 shipping. ELECTRONIC TECHNOLOGY TODAY INC., PO Box 240, Massapequa Park, New York 11762-0240.**

So there they are, two complete keyboard circuits that will latch unique keycodes from matrix and common-leg keyboards. While you could use the circuits as they are, the data they generate is raw... We still need the third basic part of any keyboard—an encoder.

The encoder

The encoder takes raw data from the keyboard and translates it into something useful—ASCII, custom data, control signals, etc. And there are lots of ways to build one. You can do it with gates, but that buys you a lot of complexity at the cost of flexibility. ROM is a much better way to go. It only takes one chip and the whole configuration of the keyboard can be changed by switching memory. Unless you've got a lot of money, the best memories to use are EPROM's. They're cheap, easy to program, and erasable.

In Fig. 6 you can see how an EPROM would be connected to the circuits we've put together. Since Chip Enable (pin 18), and Output Enable (pin 20), are both held low, a unique address on the address lines (the inputs), will result in programmed data at the outputs. And the data you put in the EPROM is whatever you want for your particular application. It can be discrete code, or, if you add a bit of clocking, whole strings of data.

Now that we know how keyboards work, what about those keyboards that are available on the surplus market? There are different types—some are ASCII encoded, some are oddball encoded, and some are not encoded at all. The data can come out of the keyboard in either parallel or serial form. You can convert between parallel and serial with only a shift register and a clock. The circuit of Fig. 7-a will convert from serial to parallel, and the circuit of Fig. 7-b will go the other way. You can build a small clock circuit or steal some pulses from the keyboard you're converting.

One last point: Although we've built our keyboards with discrete IC's, there are lots of IC's around that do all the work for you. All you do is connect the switches and add a handful of parts. Most commercial keyboards will use either this approach or do the whole job with something like an 8048 and some software. A gates-only design is a good compromise between cost and complexity. The important point is that no matter how the keyboard works, it has to have the three basic parts we covered.

R-E ROBOT

continued from page 46

The byte-wide input and output ports can also be tested. The following word tests the output latch.

```
: TEST2 BEGIN 0 150 PC! FF 150 PC!
    ?TERMINAL UNTIL ;
```

The parallel input port can be tested with the following test word. Four lines are available to you at PL1:

```
: TEST3 BEGIN 120 PC@ 10 / . CR
    ?TERMINAL UNTIL ;
```

Execute TEST3 and then short some of the inputs to ground. As you short each input, you should see the display on the screen change.

Expansion

The robot can be expanded in various ways. If your expansion project requires full use of the RPC, simply couple your circuits to the RPC bus. To interface the circuitry, you need only duplicate the wait-state generator and the bus-buffer interface described in Part 6 (May, 1987). Select a block of I/O space between Ø1ØØH and EØØØH and start designing.

If your circuit is simple and needs only one or two I/O locations connect it directly to the RERBUS, PL3. Address decoding is accomplished with a single integrated circuit and no bus drivers are needed. For simple digital inputs, digital outputs, and analog inputs, connect the circuit directly to the user connector, PL1.

Operation

Now that we have our electronics in place, it is time to consider the software required to make it all work.

The software commands to be sent to the motor control circuits should follow this sequence:

- Set up timer Ø of each 8253 (left and right wheel control) for mode 3 operation. We write control word 36H to register 3.
- Write a frequency representing a slow speed into timer Ø. We write Ø2ØØH to register Ø.
- Close the forward or reverse relay. Write 1 to location Ø12ØH.
- Now enable the PLL. Write 1 to location Ø124H.

Notice that the relays are closed before the circuit is enabled. That prevents arcing when the contacts close or open.

All those functions are programmed using RCL (*R*obotic *C*ontrol *L*anguage), a sophisticated language that is implemented in Forth. The RCL lets us control the robot's motions and functions using simple commands. Further, because Forth is extensible, RCL is extensible. That means that any code we write becomes part of the language.

That last feature is especially valuable. For instance, to control circuits connected to the RERBUS we have to change the way in which the byte store and byte fetch words operate—it's like writing new PEEK and POKE words in BASIC.

Forth's extensibility allows us to create two new words, PCX! and PCX@, that we can use to access the RERBUS. Those words will operate just like PC! and PC@ but they'll do all of the data manipulation required by the RERBUS. The computer code used to create those words is shown in Table 2.

Notice that we have documented our code with comments to allow you to determine how it operates in case something goes wrong or you want to change it. The comment immediately after the word being defined is a standard Forth-notation comment showing the effect of the word on the stack. For example, PC@ pops one argument off the stack (the address) and pushes one argument on the stack (the data). Next time, we will examine the RCL in greater depth. **R-E**

ELENCO PRODUCTS AT DISCOUNT PRICES!

TWO 100MHz SWITCHABLE PROBES INCLUDED

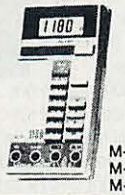
20MHz DUAL TRACE OSCILLOSCOPE
$349 MO-1251

35MHz DUAL TRACE OSCILLOSCOPE
$498 MO-1252

Top quality scopes at a very reasonable price. Contains all the desired features. Elenco's 2 year guarantee assures you of continuous service. Two 1×, 10× probes, diagrams and manual included. Write for specs.

MULTIMETER with CAPACITANCE AND TRANSISTOR TESTER
Model CM-1500 $65
Reads Volts, Ohms, Current, Capacitors, Transistors & Diodes

TRUE RMS 4½ DIGIT MULTIMETER
Model M-7000
$135
.05% DC Accuracy
.1% Resistance
with Freq. Counter
& Deluxe Case



Auto Ranging
plus
Manual Ranging
3½ Digit Meter
28 Functions
Fully protected
M-1180 .7% Acy $36.95
M-1182 .25% Acy $39.95
M-1181 .1% Acy $42.95

GF-8016 FUNCTION GENERATOR with Freq. Counter
$219

• Sine, Square, Triangle
• Pulse, Ramp, .2 to 2MHz
• Frequency .1 thru 10MHz
GF-8015 without Freq. Meter $169

10MHz OSCILLOSCOPE
$199
Model S-3000

• 10MHz DC or AC
• Triggered Sweep
• Calibrated Vert & Hor
• Reads Volts & Freq

BREADBOARD
Model 9436 Shown

9430 1,100 pins $15
9434 2,170 pins $25
9436 2,860 pins $35

DIGITAL TRIPLE POWER SUPPLY
Model XP-765
$195

0-20V @ 1A
0-20V @ 1A
5V @ 5A
Fully Regulated, Short Circuit Protected with 2 Limit Cont. 3 Separate Supplies
XP-660 with Analog Meters $149.50

DIGITAL LCR METER
$148

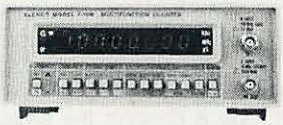
Model LC-1800
Measures: Inductors, Capacitors, Resistors

50MHz LOGIC PROBE
20 nsec with memory
LP-700

$23
Logic Pulser LP-600 $23

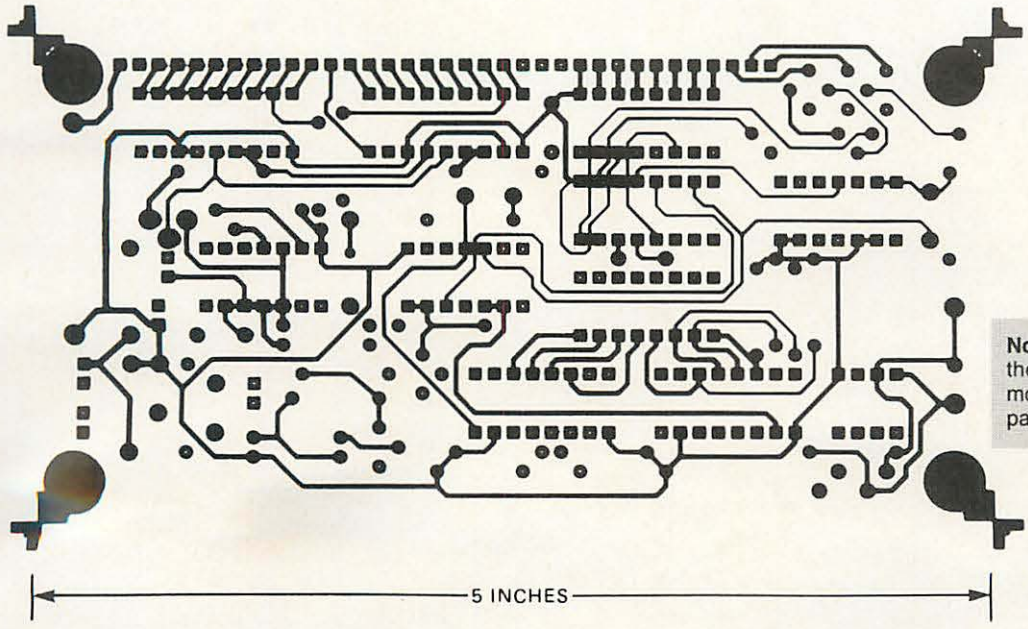
DIGITAL 3 AMP POWER SUPPLY
Model XP-750
$165

0-40V @ 1.5A
0-20V @ 3A
Fully regulated, short circuit protected current limit control
XP-650 with Analog Meters $119.50

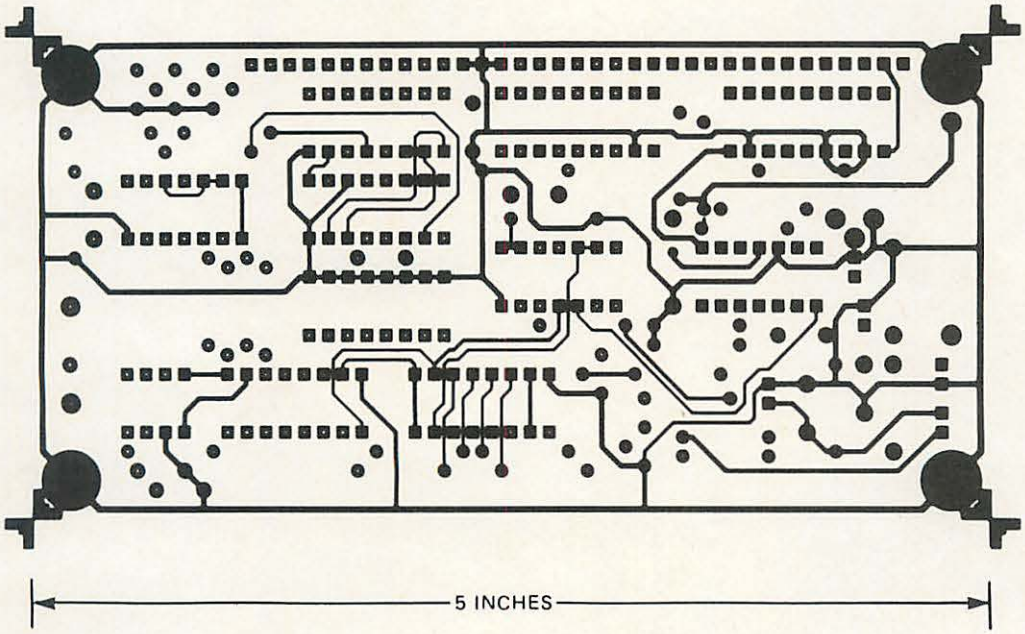
MULTI-FUNCTION COUNTERS

F-1000 1GHz $245
F-100 100MHz $169
Frequency, Period, Totalize, Self Check with High-Stabilized Crystal Oven Oscillator, 8 Digit LED Display

C&S SALES INC., 8744 W. North Ter., Niles, IL 60648
800-292-7711 (312) 459-9040 ASK FOR CATALOG
15 DAY MONEY BACK GUARANTEE
2 Year Limited Guarantee! Add 5% for Postage ($10 max), IL Res., 7% Tax
CIRCLE 109 ON FREE INFORMATION CARD

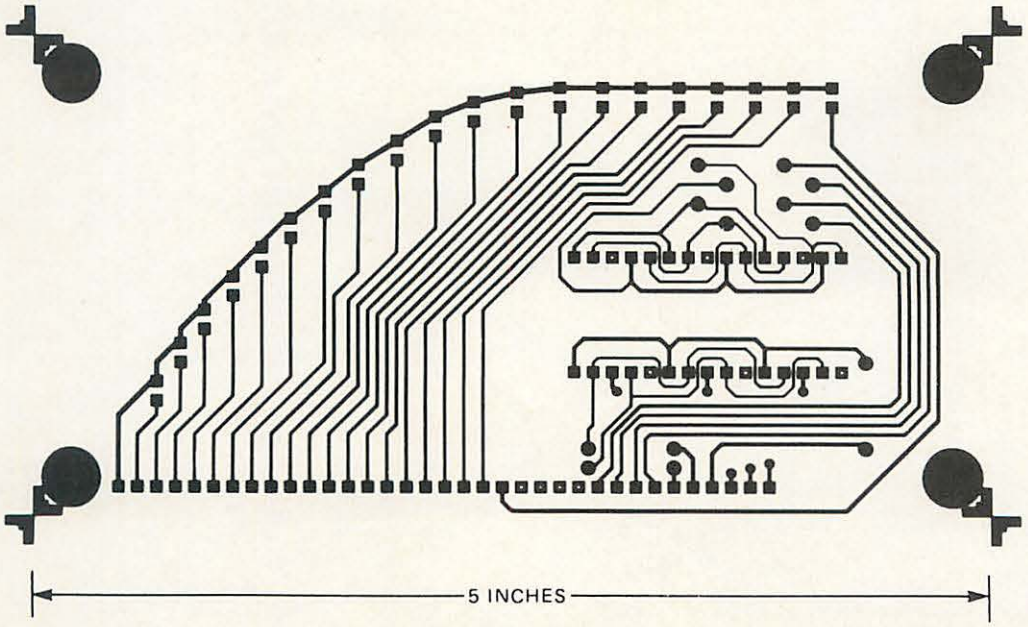
Note: The pattern for the display board of the tachometer will be presented next month along with the robot control board patterns.



THE SOLDER SIDE of the speedometer's main board is shown here.



THE COMPONENT SIDE of the speedometer's main board. When mounting components, be sure to solder all leads completely.



THE SPEEDOMETER'S DISPLAY BOARD. It is connected to the main board using 35 jumper wires.